Bifurcation analysis of 3D ocean flows using a parallel fully-implicit ocean model

Thies, Jonas; Wubs, Fred; Dijkstra, Henk A.

Published in:
Ocean modelling

DOI:
10.1016/j.ocemod.2009.07.005

Publication date:
2009

Ocean Modelling 30 (2009) 287–297

Contents lists available at ScienceDirect

Ocean Modelling

journal homepage: www.elsevier.com/locate/ocemod

Bifurcation analysis of 3D ocean flows using a parallel fully-implicit ocean model

Jonas Thies [a,*], Fred Wubs [a], Henk A. Dijkstra [b]

[a] Department of Mathematics and Computer Science, University of Groningen, Groningen, The Netherlands
[b] Institute for Marine and Atmospheric Research Utrecht, Utrecht University, Utrecht, The Netherlands

ARTICLE INFO

Article history:
Received 13 March 2009
Received in revised form 16 June 2009
Accepted 22 July 2009
Available online 29 July 2009

Keywords:
Bifurcation
Newton–Krylov methods
Implicit ocean model
Parallel implementation
Dynamical system
Trilinos

ABSTRACT

To understand the physics and dynamics of the ocean circulation, techniques of numerical bifurcation theory such as continuation methods have proved to be useful. Up to now these techniques have been applied to models with relatively few ($\mathcal{O}(10^5)$) degrees of freedom such as multi-layer quasi-geostrophic and shallow-water models and relatively low-resolution (e.g., 4° horizontal resolution) primitive equation models. In this paper, we present a new approach in which continuation methods are combined with parallel numerical linear system solvers. With this implementation, we show that it is possible to compute steady states versus parameters (and perform fully implicit time integration) of primitive equation ocean models with up to a few million degrees of freedom.

1. Introduction

Over the last decade the application of dynamical systems methods to a hierarchy of models of the ocean circulation has been a complementary approach to understand the origin of spatial-temporal variability of mid-latitude 3D ocean flows. Canonical situations are flows in a single-hemispheric spherical sector representing the North Atlantic, forced by an idealized steady double-gyre wind-stress field (with a typical amplitude τ), a restoring heat flux (with a typical pole-to-equator temperature difference ΔT), and a prescribed freshwater flux (with a typical amplitude σ). In the context of the decadal-to-interdecadal variability, there are two different limits that have been well studied. One of these limits, $\tau = 0$, represents purely buoyancy forced flows with focus on instabilities and transitions (multiple equilibria) of the meridional overturning circulation (Thual and McWilliams, 1992; Quon and Ghil, 1992, 1995; Dijkstra and Weijer, 2003). The other well-studied limit, $\Delta T = \sigma = 0$, is that of purely wind-driven flows with a prescribed density field (Dijkstra and Katsman, 1997; Primeau, 1998; Simonnet et al., 2003a,b, 2005).

Purely wind-driven stratified flows were considered in quasi-geostrophic (QG) models (Cessi and Ierley, 1995) and shallow-water (SW) models (Jiang et al., 1995). The case most relevant to the Atlantic Ocean circulation is the double gyre flow consisting of a subpolar and subtropical gyre. In QG models, when the wind stress amplitude τ is increased, multiple steady solutions appear through symmetry-breaking pitchfork bifurcations (Cessi and Ierley, 1995; Dijkstra and Katsman, 1997). Along the asymmetric solutions, a number of oscillatory instabilities occur as the lateral friction is further decreased (or wind stress is further increased); these are detected as Hopf bifurcations (Dijkstra and Katsman, 1997). In SW models, an imperfect pitchfork occurs because of the lack of a north-south symmetry and a typical bifurcation diagram (Speich et al., 1995) is sketched as the curves (i) in Fig. 1. The results provide a geometrical framework that shows the increase in complexity of the behavior of the flow when the wind-stress amplitude is increased (Simonnet et al., 2005; Pierini, 2006).

For flows with $\sigma = \tau = 0$, or purely thermally forced flows, steady states were determined versus ΔT for a $64^\circ \times 64^\circ$ sector with a constant depth of 4 km in Te Raa and Dijkstra (2002) using a primitive equation model and a $16 \times 16 \times 16$ grid. For this dynamical system with about 25,000 ($16^3 \times 5$) degrees of freedom, the steady flows (computed under restoring heat flux conditions) become unstable (under prescribed heat flux conditions) when ΔT is large enough. In this case, spontaneous multidecadal variability appears which was shown to result from a westward propagating temperature anomaly pattern that induces an out of phase response of the zonal and meridional overturning streamfunction. When the freshwater flux strength σ is increased under a restoring heat flux, the meridional overturning flow collapses (Dijkstra and Weijer, 2003). The flow changes from a northern sinking solution to a southern sinking solution. The bifurcation diagram (plotted in

* Corresponding author. Address: Department of Mathematics and Computer Science, University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK, Groningen, The Netherlands. Tel.: +31 (0)50 3636474; fax: +31 (0)50 3633800.
E-mail address: J.Thies@rug.nl (J. Thies).

1463-5003/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ocemod.2009.07.005

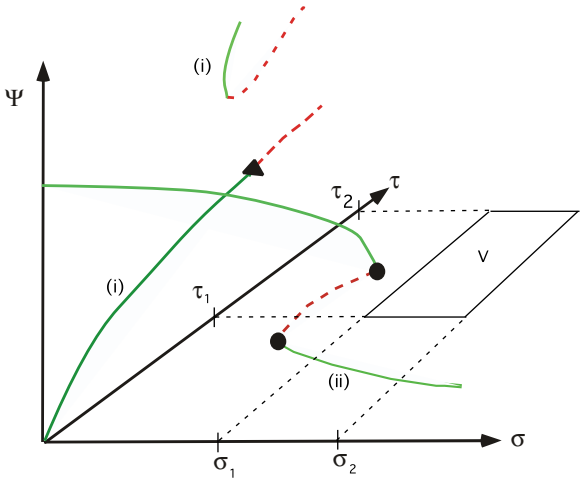

Fig. 1. Sketch of bifurcation diagrams in the different limits in the (σ,τ) plane where σ is a measure for the strength of the surface freshwater flux and τ of the wind-stress amplitude (Dijkstra, 2005); the realistic parameter regime is indicated by the area V. Green (solid) curves indicate stable steady states, while red (dashed) curves indicate unstable ones. On the vertical axis, the symbol Ψ indicates a measure of the strength of the ocean flow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 1 as the curve (ii)) consists of two saddle-node bifurcations connecting the northward overturning and the southward overturning solutions, similar to that in the two-box model (Stommel, 1961). Such a bifurcation diagram was explicitly computed for a global low-resolution ocean model (Dijkstra and Weijer, 2005) and explains the hysteresis behavior found in many ocean-climate models (Rahmstorf, 1995).

The 'realistic' parameter domain of the ocean circulation (indicated as the volume V in Fig. 1) has hardly been explored using dynamical systems methods. The reason is that one needs both a realistic size basin (to obtain an adequate representation of the meridional overturning circulation) and sufficient horizontal resolution to capture the instabilities of the wind-driven circulation. This leads to dynamical systems with a large number of degrees of freedom (e.g., a few million). All methods to tackle these problems are based on some variant of the Newton–Raphson technique. They can be distinguished into matrix-based methods (Dijkstra, 2005), where the Jacobian matrix is explicitly available and matrix-free methods, where only Jacobian matrix–vector products are needed (Knoll et al., 2005; Nadiga et al., 2006; Bernsen et al., 2008; Merlis and Khatiwala, 2008). In both matrix-based and matrix-free methods, preconditioning of the linear systems arising from the Newton–Raphson method is essential.

In this paper, we present techniques to tackle large-dimensional dynamical systems arising from relatively high-resolution primitive equation ocean models using parallel matrix-based methods. In Section 2, the ocean model formulation is briefly presented. In Section 3 the implementation of the parallel solver environment is shown and Section 4 presents results for ocean flows having a few million degrees of freedom. Our aim is to present and demonstrate that bifurcation diagrams such as Fig. 1 can be computed in the 'realistic' parameter domain. The results are summarized and discussed in Section 5, where we also will indicate how matrix-free methods may benefit from the methodology presented here.

2. The fully implicit ocean model (THCM)

For the reader's convenience, we provide a short summary of the model formulation, numerical bifurcation (continuation) techniques, implicit time integration and the parallel software environment.

2.1. Model formulation

In the THCM ocean model, we consider flows in a spherical domain bounded by longitudes ϕ_w and ϕ_e and by latitudes θ_s and θ_n with continental boundaries introduced by a land mask. The ocean basin has a bottom topography h_b and is hence bounded vertically by $z=-D+h_b(\phi,\theta)$ and a nondeformable ocean–atmosphere boundary $z=0$. The flows in this domain are forced by a heat flux Q_H (in Wm^{-2}), a zonal wind stress field (τ^ϕ,τ^θ) (in Pa) and a virtual salt flux Q_S (in ms^{-1}). Both the fluxes Q_H and Q_S are of restoring type where the surface temperature and salinity are restored to prescribed functions T_S and S_S, using restoring time scales τ_T and τ_S, respectively. The wind-stress forcing (τ^ϕ,τ^θ) is prescribed from data (Trenberth et al., 1989). Both wind and buoyancy forcing are distributed as a body forcing over the first (upper) layer of the ocean having a depth H_m.

Temperature and salinity differences in the ocean cause density differences according to the linear equation of state $\rho=\rho_0(1-\alpha_T(T-T_0)+\alpha_S(S-S_0))$, where α_T and α_S are the volumetric expansion coefficients and T_0, S_0 and ρ_0 are reference quantities. The full (dimensional) equations of the primitive equation ocean model are standard (see e.g., De Niet et al., 2007) and will not be repeated here. For the momentum mixing operators, Laplacian friction is used where A_H and A_V are the constant horizontal and vertical momentum (eddy) viscosities, respectively. Although the general tracer mixing equations (including isoneutral mixing and the Gent–McWilliams representation of eddy mixing) were formulated in De Niet et al. (2007), we will here use constant horizontal and vertical diffusivities K_H and K_V, respectively (the case $\eta_M=\eta_G=0$ in De Niet et al. (2007)). Slip conditions are assumed at the bottom boundary, while at all lateral boundaries no-slip conditions are applied.

2.2. Numerical implementation

The equations are discretized in space using a second-order accurate control volume discretization method on a staggered Arakawa B-grid in the horizontal with $i=1,\ldots,N, j=1,\ldots,M$, and a C-grid in the vertical $k=1,\ldots,L$; this combination is called a Lorenz grid. The spatially discretized model equations can be written in the form

$$\mathbf{M}\frac{d\mathbf{u}}{dt}=\mathbf{F}(\mathbf{u})=\mathbf{L}(\mathbf{u})+\mathbf{N}(\mathbf{u},\mathbf{u}), \tag{1}$$

where the vector $\mathbf{u}$ contains the unknowns (u,v,w,p,T,S) at each grid point and hence has dimension $d=6\times N\times M\times L$. The operators $\mathbf{M}$ and $\mathbf{L}$ are linear and $\mathbf{N}$ represents the nonlinear terms in the equations.

2.2.1. Continuation methods

With these methods, one aims to compute steady state solutions of the governing equations versus parameters. Steady state solutions satisfy a set of nonlinear algebraic equations of the form

$$\mathbf{F}(\mathbf{u},\mathbf{p})=0. \tag{2}$$

Here the parameter dependence of the equations is made explicit through the p-dimensional vector of parameters $\mathbf{p}$ and hence $\mathbf{F}$ is a nonlinear mapping from $\mathbf{R}^{d+p}\to\mathbf{R}^d$. To determine branches of steady solutions of the Eq. (2) as one of the parameters, say μ, is varied, the pseudo-arclength continuation method (Keller, 1977) is used. The branches $(\mathbf{u}(s),\mu(s))$ are parameterized by an 'arclength' parameter s. An additional equation is obtained by 'normalizing' the tangent

$$\dot{\mathbf{u}}_0^T(\mathbf{u}-\mathbf{u}_0)+\dot{\mu}_0(\mu-\mu_0)-\Delta s=0, \tag{3}$$

where $(\mathbf{u}_0,\mu_0)$ is an analytically known starting solution or a previously computed point on a particular branch and Δs is the step-

length. To solve the system of Eqs. (2) and (3) a predictor-corrector method is applied. The secant method and Newton–Raphson method are used as predictor and corrector, respectively.

During one Newton–Raphson iteration, with iteration index k, linear systems of the form

$$J(\mathbf{u}^k)\begin{pmatrix}\Delta\mathbf{u}^{k+1}\\ \Delta\mu^{k+1}\end{pmatrix}=\begin{pmatrix}\mathbf{r}^k\\ r^k_{d+1}\end{pmatrix}, \tag{4}$$

have to be solved, where $\Delta\mathbf{u}^{k+1}$ and $\Delta\mu^{k+1}$ are the updates, respectively. The quantities $\mathbf{r}^k$ and r^k_{d+1} are derived from (2) and (3) and are given by

$$\mathbf{r}^k=-\mathbf{F}(\mathbf{u}^k,\mu^k);\ r^k_{d+1}=\Delta s-\dot{\mathbf{u}}_0^T(\mathbf{u}^k-\mathbf{u}_0)+\dot{\mu}_0(\mu^k-\mu_0). \tag{5}$$

The $(d+1)\times(d+1)$ Jacobian matrix J of (2) and (3) along a branch is given by

$$J(\mathbf{u})=\begin{bmatrix}\Phi & \mathbf{F}_\mu\\ \dot{\mathbf{u}}_0^T & \dot{\mu}_0\end{bmatrix}, \tag{6}$$

where Φ is the matrix of derivatives of $\mathbf{F}$ with respect to $\mathbf{u}$ and $\mathbf{F}_\mu$ the derivative of $\mathbf{F}$ with respect to the parameter μ. In every continuation step, we start from a known state $\mathbf{u}_0$ and compute a tangent to the branch of solutions with respect to a chosen continuation parameter μ. Then the Newton–Raphson process is started and only systems of equations of the form

$$\Phi\mathbf{x}=\mathbf{b}, \tag{7}$$

have to be solved to compute a new point on the branch. The algorithm is sketched in Algorithms 1 and 2.

Algorithm 1. K steps of the pseudo-arclength continuation method

Input: $\mathbf{u}_0,\mu_0:\mathbf{F}(\mathbf{u}_0,\mu_0)=0, K>0$, step-size Δs
Output: $\mathbf{u}_K,\mu_K:\mathbf{F}(\mathbf{u}_K,\mu_K)=0$
for $k=0\dots K-1$ **do**
  if $k==0$ **then**
    set $\Delta\mathbf{u}=0,\Delta\mu=0$
  else
    set $\Delta\mathbf{u}=(\mathbf{u}_k-\mathbf{u}_{k-1})/\Delta s,\Delta\mu=(\mu_k-\mu_{k-1})/\Delta s$
  end
  Secant Predictor: $\mathbf{u}'=\mathbf{u}_k+\Delta s\Delta\mathbf{u},\mu'=\mu_k+\Delta s\Delta\mu$;
  Newton Corrector:
  for $i=1:it_{max}$ **do**
    Solve $\mathbf{J}(\mathbf{u}')\begin{pmatrix}\Delta\mathbf{u}\\ \Delta\mu\end{pmatrix}=\begin{pmatrix}\mathbf{r}^k\\ \mathbf{r}^k_{d+1}\end{pmatrix}$ (see Algorithm 2 and Eqs. (5) and (6));
    set $\mathbf{u}'=\mathbf{u}'+\alpha\Delta\mathbf{u},\mu'=\mu'+\alpha\Delta\mu$, where $0<\alpha\leqslant 1$ is chosen such that $||F(u',\mu')||$ decreases;
    if $||\mathbf{F}(\mathbf{u}',\mu')||$ small enough **break**;
  end
  set $\mathbf{u}_{k+1}=\mathbf{u}',\mu_{k+1}=\mu'$;
end

Algorithm 2. Solving the 'bordered system'

To solve $\begin{pmatrix}\Phi & \mathbf{F}_\mu\\ \dot{\mathbf{u}}_0^T & \dot{\mu}_0\end{pmatrix}\begin{pmatrix}\mathbf{x}_u\\ \mathbf{x}_\mu\end{pmatrix}=-\begin{pmatrix}\mathbf{f}_u\\ \mathbf{f}_\mu\end{pmatrix}$;
solve $\Phi\mathbf{a}=-\mathbf{f}_u$;
solve $\Phi\mathbf{b}=-\mathbf{F}_\mu$;
set $\mathbf{x}_\mu=-\frac{\dot{\mathbf{u}}_0^T\mathbf{a}+\mathbf{f}_\mu}{\dot{\mathbf{u}}_0^T\mathbf{b}+\dot{\mu}_0}$;
set $\mathbf{x}_u=\mathbf{a}+\mathbf{x}_\mu\mathbf{b}$;

As Φ is very large and sparse, we will use an iterative method to compute an approximate solution $\tilde{\mathbf{x}}$ in the k'th Krylov subspace

$$\mathcal{K}^k(\mathbf{x}_0)=\{\mathbf{x}_0,\Phi\mathbf{x}_0,\Phi^2\mathbf{x}_0,\dots,\Phi^{k-1}\mathbf{x}_0\},$$

where $\mathbf{x}_0$ is a starting vector. The most prominent method of this type (for non-Hermitian matrices Φ) is GMRES (Generalized Minimum Residual, Saad and Schultz (1986)). The convergence behavior of Krylov subspace methods depends on the spectral properties of the operator Φ. If Φ is diagonalized as $\Phi=Y^H\Lambda X$ (the superscript ^{H} denotes Hermitian transpose), the eigenvalues (entries in the diagonal matrix Λ) should be clustered around the value 1 for fast convergence. Furthermore, Φ should be close to a normal matrix, which means that $Y^HX\approx I$. Note that the latter property is always satisfied for Hermitian matrices, where $Y=X$.

Ill-conditioning (unfavorable spectral properties) can often be remedied by introducing a preconditioner P. The linear system (7) is replaced by the more well-behaved system

$$P^{-1}\Phi\mathbf{x}=P^{-1}\mathbf{b}. \tag{8}$$

Here P is an approximation of the matrix Φ with the properties that linear systems with P are substantially easier to solve than those with Φ and that the eigenvalues of $P^{-1}\Phi$ are more clustered around 1. Ideally, the matrix $P^{-1}\Phi$ is also closer to a normal matrix, although this is hard to verify in practice.

In the case of primitive equation ocean models, the matrix Φ is usually (very) ill conditioned and a preconditioner P must be used. When systems with the preconditioner are again solved using Krylov subspace methods, the structure of the computations is described by the following four-level scheme:

- continuation step leading (for every new parameter value) to a non-linear system of algebraic equations,
  - Newton–Raphson iteration leading to a sequence of linear correction equations,
    - outer Krylov subspace iteration for the solution of a linear correction equation,
      - inner Krylov subspace iterations to solve parts of a linear correction equation.

Our preconditioning technique will be further detailed in Section 3.1 and is summarized in Algorithms 3 and 4. The interested reader is referred to Demmel et al. (1997) or any other textbook on advanced numerical algorithms for more details on iterative solvers and preconditioning.

2.2.2. Implicit time integration

A nice spin-off of continuation methods is the immediate availability of implicit time integration schemes. Using a time step Δt, and a time index n, such a scheme can, for $\omega\in[0,1]$, be written as

$$\mathbf{M}\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}+(1-\omega)\mathbf{F}(\mathbf{u}^n,\mathbf{p})+\omega\mathbf{F}(\mathbf{u}^{n+1},\mathbf{p})=0. \tag{9}$$

For $\omega=1/2$ and $\omega=1$, these are the Crank–Nicolson method and backward Euler method, respectively (Atkinson, 1976). The equations for $\mathbf{u}^{n+1}$ are solved by the Newton–Raphson technique and lead to the same type of numerical problems as those for the steady state computation.

It is well-known that the second-order Crank–Nicolson scheme is unconditionally stable for linear equations. This does not mean that one can take any time step, since this quantity is limited by two factors. One factor is accuracy; although the scheme is second-order accurate in time, large discretization errors occur when too large time steps are used. A second limitation on the time step is the convergence domain of the Newton–Raphson process, which does not necessarily converge for every chosen time step. For many

applications, however, much larger time steps can be taken than with explicit models.

2.3. Parallelization: the Trilinos package

The Trilinos project (http://trilinos.sandia.gov) encompasses a suite of C++ libraries designed for use on parallel computers. It uses the message passing interface (MPI) to achieve distributed memory parallelism. In this section we will briefly discuss the Trilinos features relevant for this study. Apart from the packages mentioned here, several others are used for basic memory management, I/O and additional basic tasks.

- *Epetra* provides basic linear algebra classes for vectors and (sparse or dense) matrices. All other Trilinos libraries can be used with Epetra objects.
- *LOCA* is a library of continuation methods.
- *NOX* offers Newton–Raphson type solvers for large nonlinear systems of equations, reinforced by line search or trust region techniques for increased robustness.
- *AztecOO* contains Krylov subspace solvers for sparse linear systems and simple ILU preconditioners parallelized by domain decomposition.
- *Amesos* is an interface to several sequential and parallel sparse direct solvers, including *KLU* (builtin, sequential), *MUMPS* and *SuperLU*.
- *Ifpack* contains a number of sequential incomplete factorization methods (ILU, ILUT, etc.) and provides a templated class for additive Schwarz domain decomposition (overlapping and non-overlapping).
- *ML* is a parallel smoothed-aggregation multigrid library that can use (among others) *Ifpack* or *AztecOO* as smoothers and *Amesos* for the coarse grid correction.

During a continuation step, we leave the computational steering to the *LOCA* library. All we have to do is pass parameters from *LOCA* to THCM and provide the distributed THCM Jacobian Φ and residuals $\mathbf{b}$ to *LOCA*. *LOCA* then uses *NOX* to solve the nonlinear algebraic problems arising during the continuation process. *NOX* in turn uses the iterative solvers provided by *AztecOO* for what we call the 'outer iterations' with the Jacobian matrix (see the four-level scheme at the end of Section 2.2). As our preconditioner uses inner iterations, it cannot be viewed as a constant operator. For such a case one cannot use the standard GMRES but one has to resort to a flexible variant, either GMRESR (Van der Vorst and Vuik, 1994) or FGMRES (Saad, 1993).

Using the data structures provided by *Epetra*, it is comparatively easy to construct Φ and $\mathbf{b}$ in parallel such that this part of the computation is fully scalable both in CPU time and memory requirement. By first building local overlapping copies on rectangular subdomains and then 'assembling' the global linear system, it is rarely required to adjust the sequential THCM code and at no point is MPI used inside THCM. We use a 2D decomposition of the global domain into subdomains because the z-direction is typically hardly being refined whereas high resolution of the horizontal grid is desirable. This also allows a more straightforward parallelization of the block preconditioner (as described below in Section 3).

3. Block preconditioning

As indicated above, a good preconditioner is key to achieving fast convergence rates or, in our case, any convergence at all when memory and computing time are limited. In Section 3.1 we explain in short how the Jacobian matrix Φ is used to develop an efficient preconditioning matrix P. A more detailed description can be found in De Niet et al. (2007). Section 3.2 next describes the novel parallel implementation of the solution techniques using Trilinos.

3.1. The tailored preconditioner

The system (7) obtained after discretization and Newton-linearization is of the form

$$\begin{bmatrix} A_{uv} & B_1 & G_{uv} & 0 \\ 0 & 0 & G_w & B_2 \\ D_{uv} & D_w & 0 & 0 \\ B_3 & B_4 & 0 & A_{TS} \end{bmatrix} \begin{bmatrix} x_{uv} \\ x_w \\ x_p \\ x_{TS} \end{bmatrix} = \begin{bmatrix} b_{uv} \\ b_w \\ b_p \\ b_{TS} \end{bmatrix}, \tag{10}$$

The first row is the representation of the horizontal momentum equations. Here A_{uv} represents the convection–diffusion and the Coriolis operator, B_1 the influence of the vertical velocity on the horizontal momentum and G_{uv} is the horizontal gradient operator that acts on the pressure. The second row represents the hydrostatic pressure equation, so G_w is the vertical gradient operator acting on the pressure and B_2 contains the couplings to the temperature and salinity such that the combination is the density. The third row represents the conservation of mass, where D_{uv} and D_w denote the discretized divergence operator for the horizontal directions and vertical direction, respectively. Finally, the last row describes the tracer (temperature, salinity) equations. Here B_3 and B_4 determine the influence of the horizontal and vertical velocity, respectively, and A_{TS} represents the convection and mixing of heat and salt. Hence, (10) is the system for which we want to construct a preconditioner.

The approach followed is to transform the matrix in (10) into a block lower triangular form as far as possible, which then has diagonal blocks that are easy to deal with in a solution process. In this respect, it is also important to get rid of the zero blocks on the diagonal. A significant reduction can be achieved by exploiting that G_w has a large null space. In fact every pressure field that, for one cell in the horizontal plane, is constant from bottom to surface and zero elsewhere is in the null space. We split the pressure into a part in this null space, and a part which is independent of it. An orthonormal basis of the null space is denoted by M, hence $G_w M = 0$ and $M^T M = I$. Since D_w is the transpose of G_w it also holds that $M^T D_w = 0$. Now, we write $x_p = M x_{\bar{p}} + x_{\bar{\bar{p}}}$, where $x_{\bar{\bar{p}}}$ is fixed by the requirement that it is zero at the bottom. We use the bar to indicated that $x_{\bar{p}}$ is related to a depth-averaged pressure; it represents a 2D horizontal field of unknowns. The above system can be brought into the form (for details, see De Niet et al. (2007))

$$\begin{bmatrix} \widetilde{G}_w & 0 & 0 & B_2 \\ \widetilde{G}_{uv} & K_{uv\bar{p}} & \widetilde{B}_1 & 0 \\ 0 & \widetilde{D}_{uv} & \widetilde{D}_w & 0 \\ 0 & B_3 & B_4 & A_{TS} \end{bmatrix} \begin{bmatrix} x_{\bar{p}} \\ x_{uv\bar{p}} \\ x_w \\ x_{TS} \end{bmatrix} = \begin{bmatrix} b_w \\ b_{uv\bar{p}} \\ b_{\bar{p}} \\ b_{TS} \end{bmatrix}, \tag{11}$$

Here $\widetilde{G}_w$ is an adapted form of G_w in which all columns associated with the bottom pressures are left out. A system with $\widetilde{G}_w$ is easy to solve, since it consists of an independent lower-triangular system for each vertical column. In $\widetilde{G}_{uv}$ also the columns related to the bottom pressures are left out. Likewise, we leave out the associated rows from the divergences, which results in $\widetilde{D}_{uv}$ and $\widetilde{D}_w$. The omitted rows are dealt with in the new diagonal block matrix $K_{uv\bar{p}}$, defined as

$$K_{uv\bar{p}} = \begin{bmatrix} A_{uv} & \overline{G}_{uv} \\ \overline{D}_{uv} & 0 \end{bmatrix},$$

where $\overline{G}_{uv} = G_{uv}M$ and $\overline{D}_{uv} = M^T D_{uv}$. $\widetilde{B}_1$ is the trivially extended B_1 in order to accommodate for the length of the new combination $x_{uv\bar{p}}$.

The so-called block Gauss–Seidel preconditioner P is obtained by a lower-triangular approximation of the Jacobian matrix which is obtained from (11) if we omit $\widetilde{B}_1$ and B_2. The omission of $\widetilde{B}_1$ is reasonable since the influence of the vertical velocity on the horizontal momentum is quite small. However, the omission of B_2 has more severe consequences for the convergence, especially in a strong thermohaline flow, as we will see below. The application of this preconditioner requires the solution of four systems having the respective diagonal matrices as system matrix. Since the solution with $\widetilde{G}_w$ and $\widetilde{D}_w$ is trivial (both are triangular) we are left with the solution of systems with the easier systems $K_{uv\bar{p}}$ and A_{TS}. We will solve these systems approximately in every outer iteration.

To solve the linear system with $K_{uv\bar{p}}$, which is a so-called saddle-point problem, we introduce an incomplete block factorization

$$K_{uv\bar{p}} = \begin{bmatrix} A_{uv} & \overline{G}_{uv} \\ \overline{D}_{uv} & 0 \end{bmatrix} \approx \begin{bmatrix} A_{uv} & A_{uv}\widetilde{A}_{uv}^{-1}\overline{G}_{uv} \\ \overline{D}_{uv} & 0 \end{bmatrix} = \begin{bmatrix} A_{uv} & 0 \\ \overline{D}_{uv} & I \end{bmatrix} \begin{bmatrix} I & \widetilde{A}_{uv}^{-1}\overline{G}_{uv} \\ 0 & \widehat{C} \end{bmatrix}, \qquad (12)$$

where $\widehat{C} = -\overline{D}_{uv}\widetilde{A}_{uv}^{-1}\overline{G}_{uv}$ is the Schur-complement and $\widetilde{A}_{uv}$ is the 2×2 block diagonal of A_{uv}. This approximate LU-decomposition is used as a preconditioner for $K_{uv\bar{p}}$ (i.e., the modified Simple(R) method in De Niet et al. (2007)), so that the problem is further reduced to solving linear systems with A_{uv} and $\widehat{C}$.

We can now relate the solution of the Newton correction equation to the four-level scheme at the end of Section 2.2:

- Outer Krylov subspace iteration to solve system (10) using GMRESR,
  - inner Krylov subspace iteration for the saddle-point problem with $K_{uv\bar{p}}$ using GMRESR,
    - inner iteration for the systems with A_{uv} and $\widehat{C}$,
  - inner Krylov subspace iteration for the system with A_{TS} using GMRESR.

To summarize this section, we show the complete preconditioning process in Algorithms 3 and 4.

Algorithm 3. Applying the block Gauss–Seidel preconditioner

Input: right-hand side (rhs) split into components b_{uv}, b_w, b_p, b_{TS}
Output: approximate solution split into x_{uv}, x_w, x_p, x_{TS}

/* block lower triangular solve */
1 **solve** $\widetilde{G}_w \tilde{x}_p = b_w$; // triangular solve for 3D pressure
2 **set** $f_{\bar{p}} = M b_p$; // depth-average pressure
3 **set** $f_{uv} = b_{uv} - G_{uv}\tilde{x}_p$; // partial rhs for saddle-point problem
4 **solve** $\mathbf{K}_{uv\bar{p}}[x_{uv}^T\ x_{\bar{p}}^T]^T = [f_{uv}^T\ f_{\bar{p}}^T]^T$; // PGMRESR, Algorithm 4 as prec.
5 **set** $x_p = \tilde{x}_p + M^T x_{\bar{p}}$; // final 3D pressure field
6 **set** $f_w = b_p - \widetilde{D}_{uv}x_{uv}$; // rhs for vertical velocity w
7 **solve** $\widetilde{D}_w x_w = f_w$; // diagnose w from continuity
8 **set** $f_{TS} = b_{TS} - B_3 x_{uv} - B_4 x_w$; // rhs for tracer equations
9 **solve** $A_{TS}x_{TS} = f_{TS}$; // using preconditioned GMRES[1]

[1] cf. Section 3.3.

Algorithm 4. Modified simple method for $K_{uv\bar{p}}$ (cf. Eq. (12))

Input: right-hand side vector split into components $b_{uv}, b_{\bar{p}}$
Output: approximate solution split into $x_{uv}, x_{\bar{p}}$

/* 1) block lower triangular solve */
1 **solve** $A_{uv}y_{uv} = b_{uv}$; // using preconditioned GMRES[1]
2 **set** $y_{\bar{p}} = b_{\bar{p}} - \overline{D}_{uv}y_{uv}$;
/* 2) block upper triangular solve */
3 **solve** $\widehat{C}x_{\bar{p}} = y_{\bar{p}}$; // using an LU decomposition[1]
4 **set** $x_{uv} = y_{uv} - \widetilde{A}_{uv}^{-1}\overline{G}_{uv}x_{\bar{p}}$

3.2. Implementation in Trilinos

The most challenging aspect of implementing a fully implicit ocean model like THCM on a parallel platform is unquestionably the preconditioner. We will therefore present in some detail a Trilinos adaptation of the block preconditioner discussed in Section 3.1 for use on distributed memory machines.

In order to precondition the saddle-point problem $K_{uv\bar{p}}$, we use the modified Simple(R) method as discussed above. It factors the 2×2 block matrix approximately such that the two easier linear systems A_{uv} and $\widehat{C}$ have to be solved, each twice per iteration. The method then requires the following steps:

- Setup:
  - Extract and re-index the various submatrices. This can be done using basic *Epetra* operations, yielding the submatrices distributed among processors according to the decomposition of the Jacobian (fully scalable in memory and CPU time).
  - Construct the preconditioner 'hardware', i.e., depth-averaging operator M and $\overline{G}_{uv} = G_{uv}M, \overline{D}_{uv} = M^T D_{uv}$. This has to be done only once for the entire continuation process as the corresponding terms are linear and constant w.r.t. the continuation parameters.
  - Construct the Simple(R) preconditioner, in particular the Schur-complement $\widehat{C} = \overline{D}_{uv}\widetilde{A}_{uv}^{-1}\overline{G}_{uv}$, where $\widetilde{A}_{uv}$ is the 2×2 block diagonal of the convection–diffusion/Coriolis operator A_{uv}. This matrix-matrix product is cheap in practice as the components are very sparse and small because of the depth-averaging.
  - Construct preconditioners for $A_{uv}, \widehat{C}$ and the tracer system A_{TS}. This issue will be further detailed in the next section.
- Apply inverse: Once the matrices are available in distributed form and parallel preconditioners have been built, the lower triangular solve can be done the same way as in the sequential case. A decision has to be made how accurately the two linear systems with $K_{uv\bar{p}}$ and A_{TS} should be solved. At higher resolution it is typically advisable to do a few GMRES iterations for both problems so that an accuracy of $10^{-4} - 10^{-2}$ is achieved.

The crucial operations in the above procedure are

- sparse matrix–vector products (for constructing the Krylov subspaces),
- setup of efficient algebraic preconditioners for $A_{uv}, \widehat{C}$ and A_{TS},
- applying the (inverse) preconditioners to a vector.

Assuming that the first is provided by *Epetra*, we will now turn to the issue of parallel algebraic preconditioners.

3.3. Algebraic preconditioners

The block preconditioning strategy requires three different linear algebraic systems to be solved. The system matrices are A_{uv} (convection/diffusion/Coriolis), $\widehat{C}$ (Schur-complement of Simple(R) method), and A_{TS} (tracer advection/diffusion and mixing). All of these matrices are sparse and non-symmetric, so preconditioned GMRES is a good choice. We seek preconditioners that are so effective that only very few or no inner iterations are required in order to keep the cost per outer iteration at bay.

Among parallel preconditioners, domain decomposition methods like the additive Schwarz method are a class of simple yet effective algorithms for distributed memory machines. The idea is to use a good sequential approximate or direct solver on each subdomain independently. The subdomains typically correspond to data in the physical memory of the respective processors. To improve the parallel performance, overlap between the subdomains can be introduced. However, better parallel performance is typically achieved by introducing a coarse grid correction. A detailed description of such two- or multi-level methods would go beyond the scope of this paper, so for details we refer the reader to Tuminaro et al. (2005) and references therein. The basic idea is to introduce a related problem on a coarser grid and pass iteratively between the uncoupled solver ('smoother') on the fine level and a direct solver for the coarse problem. This has two major benefits compared to an uncoupled approximate solver on the fine level only. Firstly, smooth error components are quickly eliminated by the coarse grid correction. Secondly, the coarse direct solve makes up for the connections between the subdomains which are lost in the domain decomposition process. This is important to maintain good performance as the number of subdomains increases.

We use the Trilinos package *ML* as a framework for constructing a hierarchy of two or more grids. On the coarsest level we use a direct sparse solver, typically the sequential KLU solver from the *Amesos* library. For the fine level(s) we use the additive Schwarz method as implemented in *Ifpack* or *AztecOO*, either without or with a small amount of overlap. On each subdomain we can then use a sequential approximate solver, for instance an incomplete LU factorization. A particular choice of subdomain solver is MRILU (Botta and Wubs, 1999), as it has been developed with multi-scale CFD applications in mind and has been used successfully in the sequential case (De Niet et al., 2007). In the multigrid context of *ML*, it is important that the subdomain solver has a smoothing effect on the residual on the respective level. To this end, lumping strategies like the Gustafsson modification should be disabled in MRILU. We give the details on how the systems associated with the indicated matrices are solved below.

$\mathbf{A_{uv}}$: The matrix denoted by A_{uv} yields a relatively easy linear system. It is, however, important that the 2×2 block diagonal is well-represented in the preconditioner so that the Coriolis term is captured, which contributes significantly to the operator in applications with a relatively large domain. In practice a three-level hierarchy using the 'MIS' aggregation scheme in *ML* (Tuminaro et al., 2005) has turned out to give a very good preconditioner. As smoother on the fine and intermediate levels, we use the *AztecOO* implementation of additive Schwarz with a zero-fill ILU on each subdomain (the *AztecOO* implementation of ILU(0) seems to be faster than the *Ifpack* version, as of Trilinos 9.0). A single multigrid cycle is typically sufficient to ensure fast convergence of the Simple(R) method. The small number of levels leads to a low operator complexity (i.e., few additional unknowns in the newly introduced coarse problems), and the ILU(0) smoothing is very fast both in setup and application. An alternative to the ILU(0) would be block Gauss–Seidel with blocksize 2 to capture the Coriolis term.

$\widehat{\mathbf{C}}$: The matrix denoted by $\widehat{C}$ in Section 3.1 corresponds to a 2D ('depth-averaged') grid with one unknown per grid cell (the pressure). In a simulation with 16 depth layers, it is therefore about 100 times smaller than the original matrix (16×6). We can therefore use a direct sparse solver for this system, which gives excellent convergence rates for the Simple(R) method on the saddle-point problem. For the moderately sized problems we have treated so far (two million unknowns on 32 CPU's, cf. Section 4) it is most efficient to use the sequential KLU solver. As the number of processes and grid cells increases, one might want to use a distributed memory solver on a subset of the processors or a two-level technique with KLU on the subdomains and coarse grid. Both approaches have been verified to work but are less efficient than the sequential KLU for the cases presented here.

$\mathbf{A_{TS}}$: The most challenging matrix encountered in our block preconditioner is the one denoted by A_{TS}. It represents advection and diffusion of heat and salt, as well as subgrid scale mixing and convective adjustment. The convective adjustment procedure in THCM is implemented in terms of the orthogonal variables $\rho = \lambda S - T$ (density) and $\rho^{\perp} = \lambda S + T$. Consequently, couplings between T and S in adjacent grid cells are introduced in the matrix, which creates very large off-diagonal coefficients in parts of the domain. In order to alleviate this problem, we introduce an orthogonal block scaling of the form

$$S = \mathbf{diag}\left[\frac{1}{\sqrt{2}}\begin{bmatrix} -I & \lambda \\ \frac{1}{\lambda}I & I \end{bmatrix}\right], \tag{13}$$

and solve the rescaled system

$$\underbrace{SA_{TS}S}_{A_{\rho\rho^{\perp}}}\underbrace{Su_{TS}}_{u_{\rho\rho^{\perp}}} = \underbrace{Sf_{TS}}_{f_{\rho\rho^{\perp}}},$$

instead, in which the new variables ρ and $\rho^{\perp}$ are decoupled in regions with convective adjustment and weakly coupled in stably stratified regions.

We use a two-level method to solve the system with $A_{\rho\rho^{\perp}}$. On the fine level we use non-overlapping Schwarz with MRILU on each subdomain. In order to maintain sparsity, we use a relative drop tolerance of $\epsilon_w = 0.1$ in MRILU which gives again very low setup and application costs in terms of both CPU time and memory. A coarse grid operator is constructed using the METIS (Karypis and Kumar, 1998) aggregation strategy implemented in *ML*. This allows to quickly create large aggregates and leads to much stronger coarsening than the MIS strategy. It also gives more control over the size of the aggregates and thus over the operator complexity and efficiency. In order to get acceptable convergence rates in the outer GMRESR loop, we use a short Krylov sequence when solving $A_{\rho\rho^{\perp}}$, typically at most 10 iterations unless a relative residual norm of $||\mathbf{r}||/||\mathbf{r}_0|| < 10^{-3}$ is reached first.

4. Numerical results

To demonstrate the capabilities and efficiency of the parallel implicit ocean model, we will present results of two test problems. The first test problem is the same as in De Niet et al. (2007) for which model solutions can now be obtained with much higher resolution. This case also serves as a test of the performance of our new parallel implementation. To demonstrate the practical relevance of the techniques, the second test problem is a realistic Atlantic configuration with full bathymetry and realistic forcing.

The computations were performed on a 16 CPU shared memory node of the Huygens supercomputer at the Academic Computer Center in Amsterdam (`http://www.sara.nl`). Each CPU consists of two IBM Power 6 cores (4.7 GHz), so that a maximum of 32 processes can be used efficiently. We do not make use of the system's

possibility to run two threads per core as our application is memory intensive, so additional speed-up is not expected. It should be mentioned that it is not necessary to have a shared memory machine. In fact, the same computations were done using CPUs on several Huygens nodes and only a slight performance drop was observed (this aspect was not investigated systematically, however) because of the extra communication via the network.

4.1. An idealized problem

As in De Niet et al. (2007), the ocean domain is chosen to be $\phi \in [276, 350]$, $\theta \in [10, 74]$ and $z \in [-4000, 0]$m and it has a flat bottom. In the heat flux and freshwater flux forcing, the surface temperature and surface salinity are restored to prescribed profiles given by

$$T_S(\phi,\theta) = T_0 + \eta \frac{\Delta T}{2} \cos\left(\pi \frac{\theta - \theta_s}{\theta_n - \theta_s}\right), \tag{14a}$$

$$S_S(\phi,\theta) = S_0 + \eta \frac{\Delta S}{2} \cos\left(\pi \frac{\theta - \theta_s}{\theta_n - \theta_s}\right), \tag{14b}$$

with $\Delta T = 20\,^{\circ}$C and $\Delta S = 1$ psu, using a restoring time scale of 75 days. In (14b), we have introduced a homotopy parameter η such that $\eta = 0$ represents no forcing and $\eta = 1$ is a realistic strength of the forcing. As wind forcing, interpolated values of the annual mean wind stress data from Trenberth et al. (1989) are used. The amplitude of the wind stress is also multiplied by η. In this way, we can vary η by continuation and affect all three forcing mechanisms simultaneously. Other parameters are chosen as $\alpha_T = 10^{-4}\ \mathrm{K}^{-1}, \alpha_S = 7.6 \times 10^{-4}, A_H = 2.5 \times 10^5\ \mathrm{m^2\ s^{-1}}, A_V = 10^{-3}\ \mathrm{m^2\ s^{-1}}, K_H = 10^3\ \mathrm{m^2\ s^{-1}}$ and $K_V = 1.0 \times 10^{-4}\ \mathrm{m^2\ s^{-1}}$.

4.1.1. Basic bifurcation diagram

For this problem, we start with the trivial solution for $\eta = 0$ (no forcing) and continue the steady solutions of the model in the parameter η up to $\eta = 1$ (full forcing). In De Niet et al. (2007), results were shown up to a model resolution of $64 \times 64 \times 16$ using the serial implementation of THCM. In the parallel implementation here, we use a resolution of $128 \times 128 \times 16$, yielding a dynamical system of 1,572,864 degrees of freedom.

The maxima of the meridional overturning streamfunction (ψ_M), the positive maximum of the barotropic streamfunction (subtropical gyre, ψ_T) and the negative minimum of barotropic streamfunction (subpolar gyre, ψ_P) are shown as functions of η in Fig. 2a. The strength of the gyre circulation increases approximately linearly with η, whereas ψ_M follows a power law increase. The patterns of the meridional overturning streamfunction and the barotropic streamfunction at $\eta = 1$ (full forcing) are plotted in Fig. 2b and c, respectively. These are very similar to those in De Niet et al. (2007), with a single overturning cell and a double-gyre horizontal flow pattern.

4.1.2. Performance

The idealized case serves to study the performance of the methodology and parallel implementation. We focus on a single continuation step of $\Delta s = 5 \times 10^{-3}$ in the forcing parameter η starting from the solution for $\eta = 1$ (as shown in Fig. 2b and c). For this case, the Newton–Raphson method converges in three iterations, the non-restarted (classical) GMRESR solver requires an average of 140 iterations for the solution of the Newton correction equation and a maximum of 5 GMRESR iterations is allowed to reach a relative tolerance of 10^{-3} for the saddle-point problem (no inner iterations on A_{uv} or $\widehat{C}$). A maximum of 10 GMRESR iterations is allowed for A_{TS} to reach the same relative tolerance. On the Huygens computer the single continuation step takes about two hours of wallclock time on four CPU's. Because of memory limitations it is not possible to run this computation sequentially.

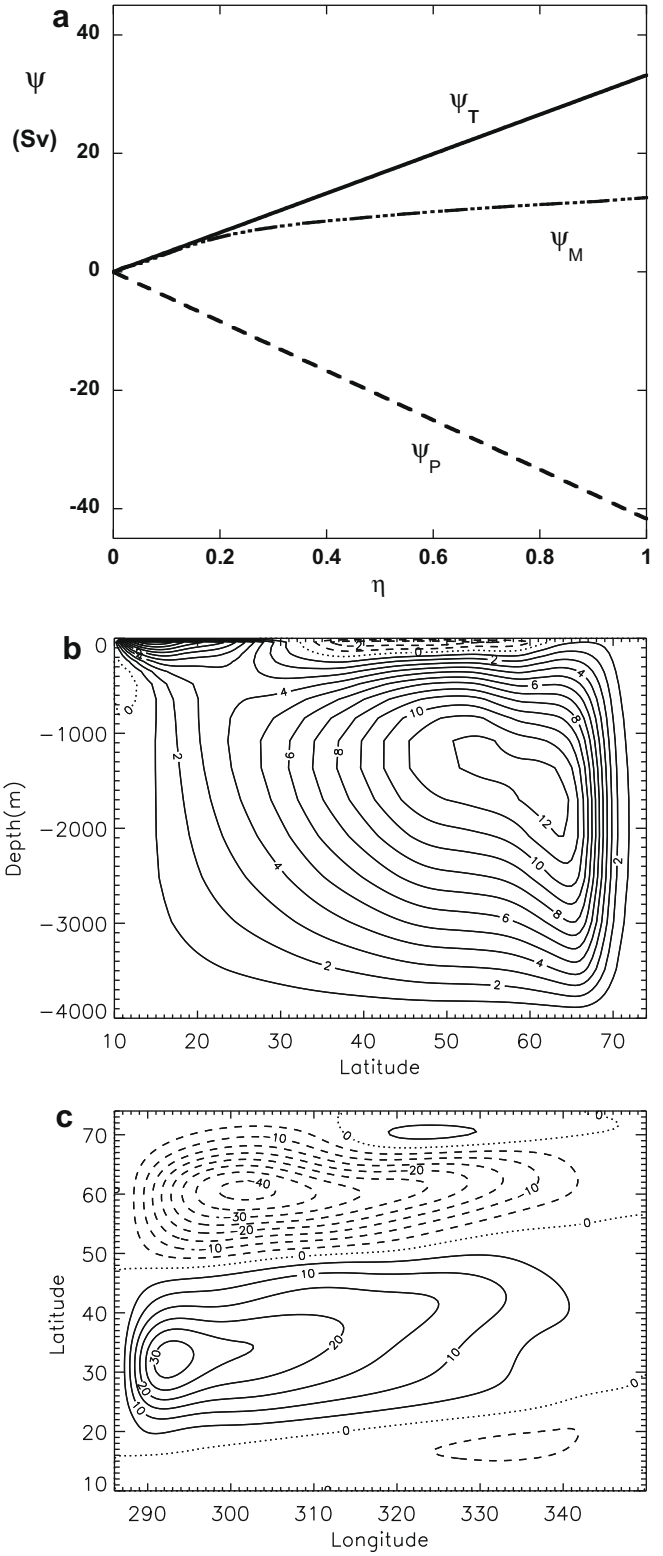

Fig. 2. (a) Maximum of the meridional streamfunction (ψ_M), the subtropical gyre barotropic streamfunction (ψ_T) and negative minimum of the subpolar gyre barotropic streamfunction (ψ_P) versus η. (b) Contour plot of the meridional overturning streamfunction for $\eta = 1$. (c) Contour plot of the barotropic streamfunction for $\eta = 1$.

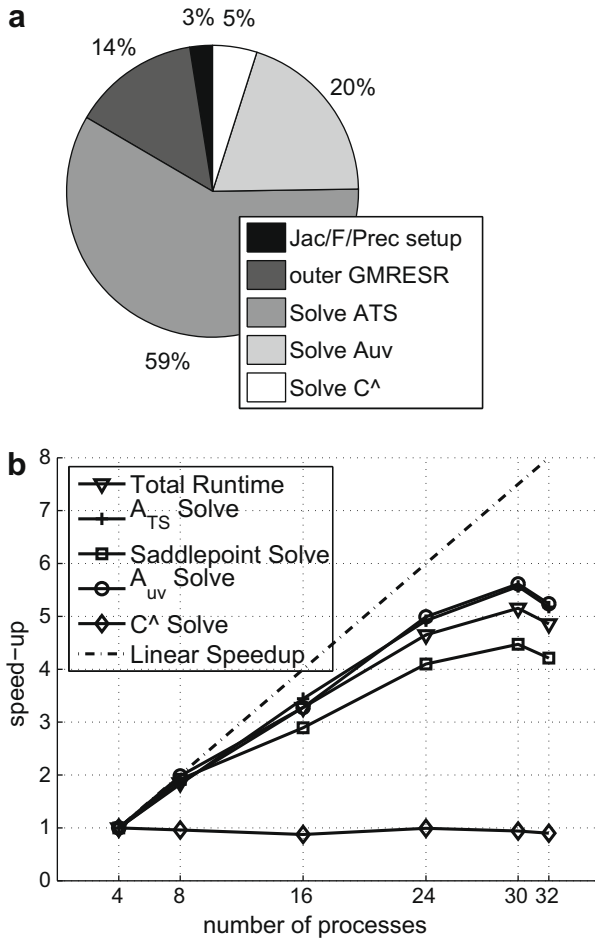

Fig. 3. (a) Profiling of the parallel code for the case of 16 MPI tasks. (b) Speed-up for various subroutines.

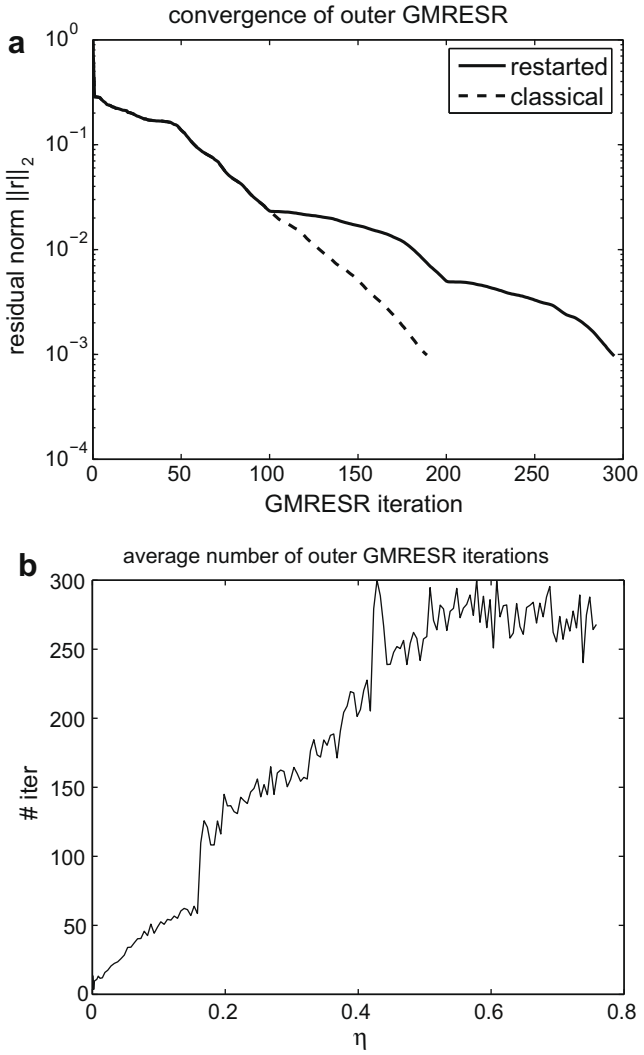

Fig. 4. (a) Performance of the classical (non-restarted) and restarted GMRESR method. (b) Number of GMRESR iterations versus the homotopy parameter η.

Fig. 3a shows in which parts of the code time is spent in the case of 16 MPI tasks. Computing the Jacobian and residual (right-hand side) is fully parallel and can therefore be expected to contribute little as the number of tasks increases. The fact that setting up the preconditioner hardly contributes either is remarkable, though, as this used to be a major part of the runtime in the sequential case. It can be explained by the use of multi-level techniques for A_{TS} and A_{uv}, which allow more relaxed drop tolerances for the smoothers on the fine level. The largest amount of time by far is spent in the GMRESR loop and during the solution of the subsystems with the matrices A_{uv}, A_{TS} and $\widehat{C}$. Fig. 3b shows the speed-up for this experiment, split up into the crucial subsystems that have to be solved. Note that a sequential solver is used for $\widehat{C}$ so that speedup is not to be expected. The saddle-point solve includes both the solution of A_{uv} and $\widehat{C}$. The performance drop at 32 tasks is explained by the fact that at least one core per node typically runs system tasks like daemons, etc., leading to a load imbalance.

4.1.3. Numerical aspects

From a numerical point of view, the restarted GMRESR method used for our outer iterations is probably not the best choice. As can be seen in Fig. 4a, the number of iterations required in a typical solve increases by about 50% by using a simple restarting technique only twice. On the other hand, the non-restarted GMRESR method is not much faster because the size of the subspace becomes quite large. An idea to improve this is to employ recently developed restarting techniques (Morgan, 2000). Another interesting approach is subspace recycling as in the GCROT method (Parks et al., 2006). Such methods make use of subspaces from previous Newton–Raphson or even continuation steps to reduce the number of outer Krylov subspace iterations.

Another observation concerns the preconditioner. The block Gauss–Seidel preconditioner used here has one major deficit: the buoyancy term B_2 is not represented. This causes the number of iterations required by the outer GMRESR loop to increase unduly as the temperature and salinity forcing grow stronger. This is illustrated in Fig. 4b, where the average number of linear solver iterations per continuation step is displayed against the continuation parameter η. Note that a part of this increase is likely caused by increased convection (here represented by convective adjustment) and cannot be avoided. In order to improve the performance for buoyancy-dominated flows, one might use a type of over-relaxation like the block-SOR method instead of standard Gauss–Seidel. Another idea is to abandon the block solver for an algebraic solver similar to the ML-MRILU method used to precondition A_{TS} in this study.

Finally, the *NOX* nonlinear solver allows us to choose a fixed Newton convergence tolerance and then select the convergence criterion for the outer Krylov subspace method depending on the accuracy of the current approximation, which prevents 'over-' or 'under-solving' the linear systems. A simple yet effective improvement to our preconditioner would be to take this idea into the

inner iterations and choose the convergence tolerances depending on the residual norm of the outer Krylov subspace method.

4.2. A more realistic problem

To compute steady flows versus parameters in a realistic Atlantic geometry, we choose the domain to be $\phi \in [262, 350], \theta >$and $z \in [-4000, 0]$ m, with full bathymetry from the ETOPO10 data set. Annual mean wind-stress forcing is again taken from data of Trenberth et al. (1989). Furthermore, the restoring profiles of T_S and S_S are taken from the Levitus (1994) data set and a restoring time scale of 75 days is used. Again, the restoring temperature and salinity profiles, as well as the wind stress amplitude are multiplied by the homotopy parameter η. Other parameters are chosen as $\alpha_T = 1.8 \times 10^{-4}$ K^{-1}, $\alpha_S = 7.6 \times 10^{-4}$, $A_H = 10^5$ m^2 s^{-1}, $A_V = 10^{-3}$ m^2 s^{-1}, $K_H = 10^3$ m^2 s^{-1} and $K_V = 1.0 \times 10^{-4}$ m^2 s^{-1}.

For these computations, a $176 \times 128 \times 16$ grid (corresponding to 1/2° horizontal resolution) is used giving a dynamical system with 2,162,688 degrees of freedom. First, the homotopy parameter η is increased from $\eta = 0$ (zero forcing) to $\eta = 1$ (realistic forcing) giving the results in Fig. 5. The maximum of the meridional overturning again increases with η following a power law (Fig. 5a). The pattern of the meridional overturning streamfunction in Fig. 5b indicates a single overturning cell, but there are regions of large gradients due to a combination of the Levitus forcing, a linear equation of state and the bottom topography. While the strength of the subtropical gyre still increases approximately linear with η, the subpolar gyre strength depends nonlinearly on η and appears to saturate at a value just above 10 Sv (Fig. 5a). The pattern of the barotropic streamfunction in Fig. 5c indeed shows a very confined subpolar gyre and a much larger subtropical gyre with a flow near the western boundary qualitatively resembling the Gulf Stream.

The solution at $\eta = 1$ (Fig. 5b and c) is also used to demonstrate the capabilities of the implicit time integration using the same model. Starting from the initial motionless solution (with homogeneous temperature and salinity field) and $\eta = 1$, the model was integrated in time using the fully-implicit Crank–Nicolson scheme ($\omega = 0.5$ in Eq. (9)). Initially, values of the three streamfunction extrema rapidly increase (Fig. 6a) but eventually settle down towards steady state values as in Fig. 5a.

The most important result here is in the time steps which can be used during the integration (Fig. 6b). The time step is limited here by the convergence of the Newton process and was adapted as follows: when the Newton–Raphson process did not converge within 10 iterations, the time step was halved, when it converged three times in a row within 8 or fewer iterations, the time step was multiplied by a factor 1.25. While the solution changes rapidly small time steps must be taken, but in the approach to steady state time steps up to 100 years can be taken (Fig. 6b).

As a first step towards computing bifurcation diagrams at this resolution, we determine one branch of solutions in the parameter A_H. The value of $A_H = 10^5$ m^2 s^{-1} is much used in relatively low-resolution ocean models, such as in De Niet et al. (2007), but can be taken smaller here. A lower bound on A_H can be estimated from the Munk layer thickness, $\delta_M \sim (A_H/\beta)^{1/3}$, where β is the local derivative of the Coriolis parameter. To resolve the western boundary current, $\Delta x = r_0 \cos\theta \Delta\phi$ (where r_0 is the Earth's radius) should be smaller than δ_M. In our case of a horizontal resolution of 1/2°, this suggests $A_H > 3000$ m^2 s^{-1}. Better results are achieved when there are several grid cells in the Munk layer, so we choose a target value of $A_H = 10^4$ m^2 s^{-1}. The nice element of continuation is that one can directly determine steady states versus the parameter A_H.

Results on the maximum value of the meridional overturning streamfunction ψ_M and the strength of both subtropical and subpolar gyre versus A_H are provided in Fig. 7a. As expected the value of ψ_M does not change much over this range of A_H, and the gyres get only stronger at lower end values of A_H. Here, the more inertial regime is entered and based on the results in Schmeits and Dijkstra (2001) with a shallow water model, it is here that multiple equilibria are expected (we did not determine the other solution branch yet). The meridional overturning pattern in Fig. 7b only changes

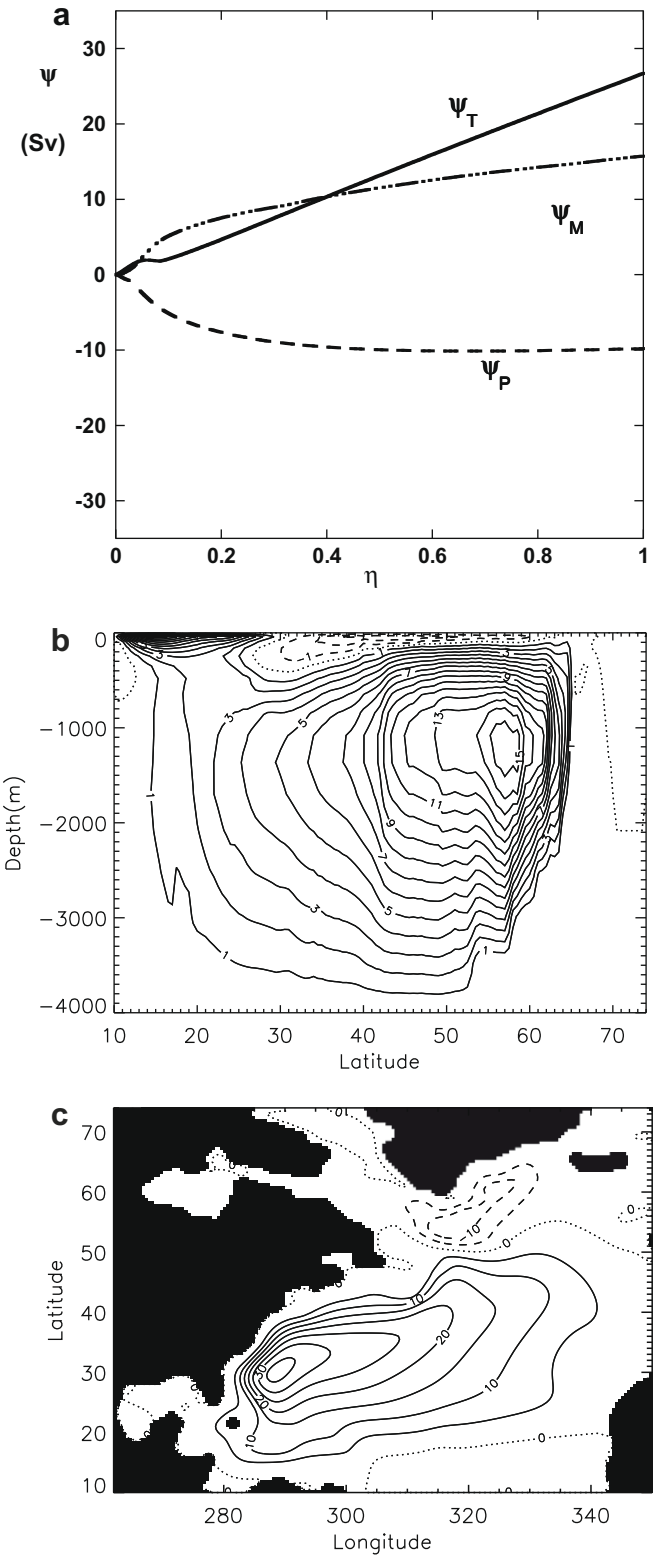

Fig. 5. (a) Values of ψ_M, ψ_T and ψ_P (as in Fig. 2) versus η. (b) Contour plot of the meridional overturning streamfunction for $\eta = 1$. (c) Contour plot of the barotropic streamfunction for $\eta = 1$.

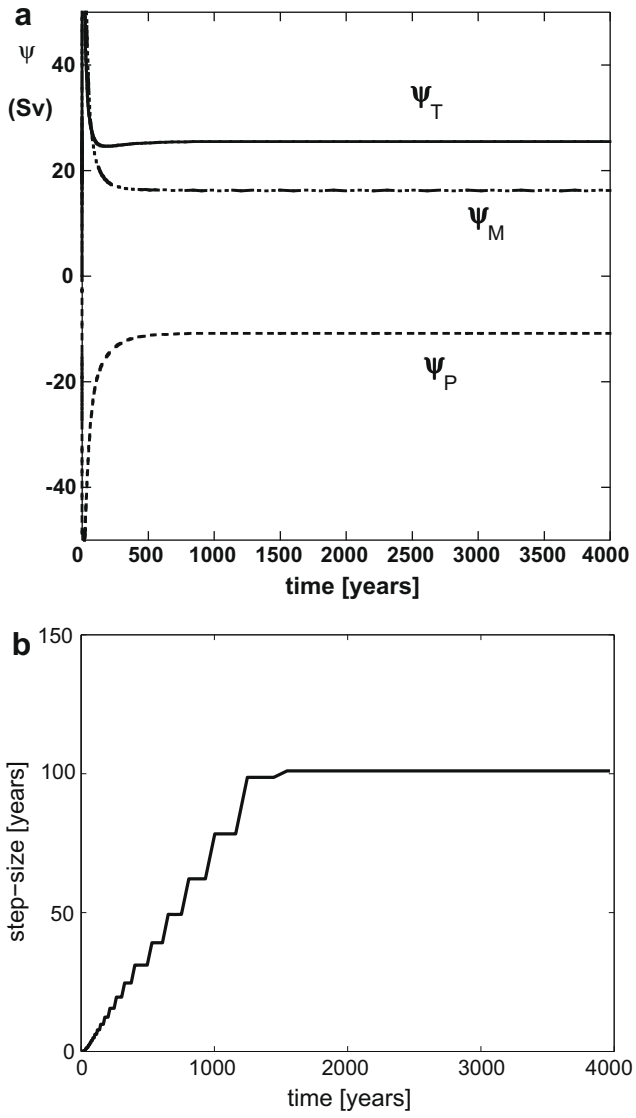

Fig. 6. (a) Values of ψ_M, ψ_T and ψ_P versus time for $A_H = 10^5$ m^2 s^{-1} (this simulation was performed with an older version of the code and slightly different parameter settings, which is why the values are not exactly the same as in Fig. 5a). (b) Time steps used along the integration path.

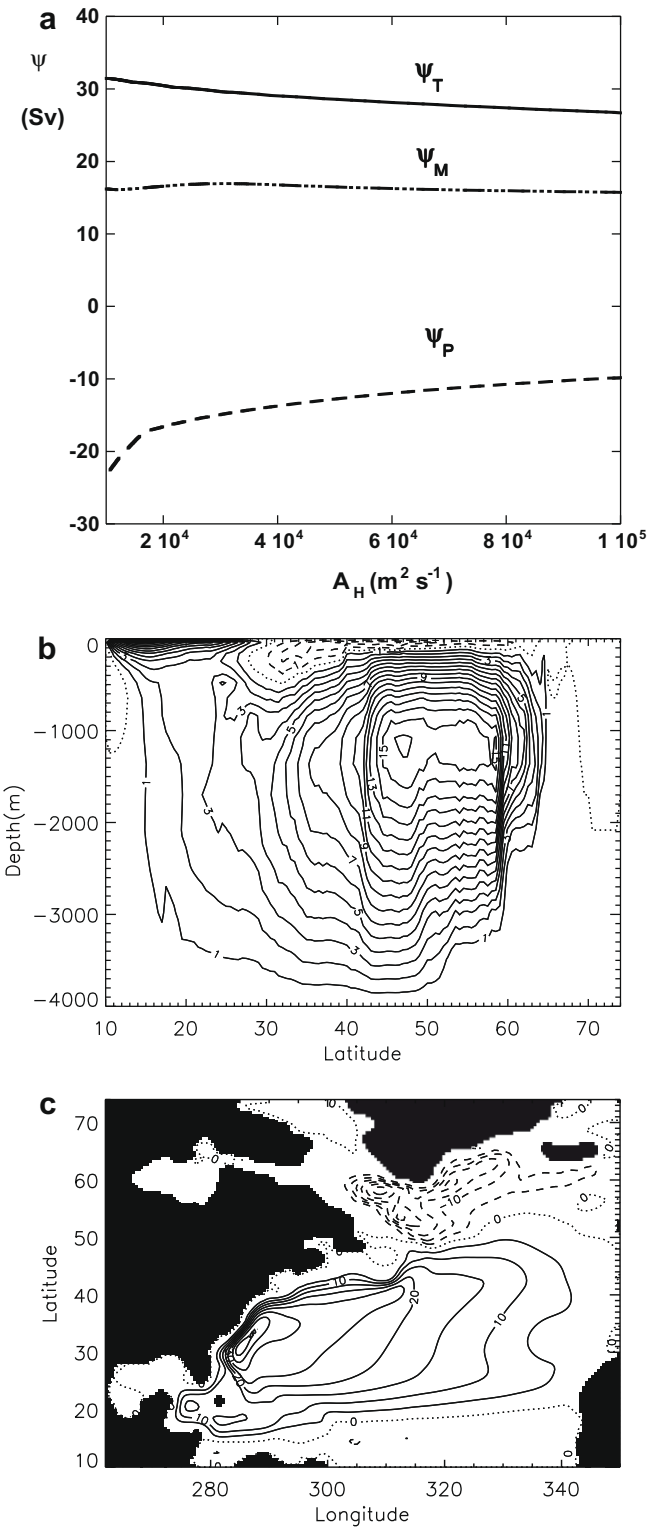

Fig. 7. (a) Values of ψ_M, ψ_T and ψ_P (as in Fig. 2) versus the horizontal eddy-viscosity A_H. (b) Plot of the meridional overturning streamfunction for $A_H = 10^4$ m^2 s^{-1}. (c) Plot of the barotropic streamfunction for $A_H = 10^4$ m^2 s^{-1}.

slightly with respect to that in Fig. 5b, but the western boundary current is much more confined to the coast in Fig. 7c than in Fig. 5c, reflecting the higher inertia of the flow.

5. Summary and discussion

So far, bifurcation analyses of ocean models were limited to relatively low resolution situations because of the difficulty of using a parallel implementation and in particular parallel preconditioners. The main contribution of this work is to demonstrate that due to the coupling of the fully implicit ocean model (THCM) to the Trilinos software, it is possible to perform bifurcation analyses with primitive equation ocean models that have a few million degrees of freedom. In addition, we have presented a novel parallel implementation of a tailored linear solver for fully implicit ocean models and studied its performance.

The results for the idealized case in Section 4.1.2 showed that the overall speed-up of the parallel fully implicit ocean model is quite satisfying (note that for sequential computations the algorithm is rather similar to the one presented in De Niet et al. (2007) and can be expected to perform roughly as good). In Section 4.1.3, we investigated 'strong' scalability, i.e., an increasing number of processes solving a problem of fixed size. We are confident that the performance on larger problems will be better and scale to larger numbers of tasks because of the better ratio of computation to communication. As already indicated in Section 4.1.3 there are, however, clearly a number of points which can be improved.

With the second test problem, we have demonstrated that with the parallel approach, one is now able to compute bifurcation diagrams for wind- and buoyancy forced 3D realistic ocean models for reasonable horizontal resolutions (here we used 1/2° resolution in an Atlantic size basin). Although we focused on only one branch of solutions with the horizontal mixing A_H as a typical control parameter, the methods provide the machinery to study multiple equilibria in the combined wind- and thermohaline circulation and to tackle the problems put forward in the introduction. We also showed that one can efficiently perform implicit time stepping, with time steps of up to 100 years in the approach towards steady states.

Apart from its importance in allowing to compute steady states versus parameters, the development of the parallel block preconditioner will also be important for further application of the matrix free methods, such as the Jacobian Free Newton–Krylov (JFNK) methods (Knoll et al., 2005; Nadiga et al., 2006; Bernsen et al., 2008, 2009; Merlis and Khatiwala, 2008). While the JFNK methods may in principle have a broader applicability as they can be applied to existing explicit time dependent ocean models, at some point efficient preconditioners are also needed to solve the systems of equations arising from the Newton–Raphson method. To integrate the matrix-based and matrix-free methodologies, we are now starting to implement the parallel preconditioner (described in this paper) into the POP ocean model with the final aim to reduce the spin-up time in the POP model. When this is successful, the investment into the development of the implicit methods will finally be available (and very likely beneficial) to a large group of ocean modelers.

Acknowledgements

Computations were done on the Huygens IBM p6 supercomputer at SARA Amsterdam. Use of these computing facilities was sponsored by the National Computing Facilities Foundation (N.C.F.) under the project SH115-08 with financial support from the Netherlands Organization for Scientific Research (N.W.O.). We thank Andy Salinger (Sandia National Labs, USA) for his help with the Trilinos software. This project was funded by NWO through the contract ALW854.00.028.

References

Atkinson, K.E., 1976. An Introduction to Numerical Analysis. John Wiley and Sons.
Bernsen, E., Wubs, F.W., Dijkstra, H.A., 2008. A method to reduce the spin-up time of ocean models. Ocean Modell. 20, 380–392.
Bernsen, E., Dijkstra, H.A., Wubs, F.W., 2009. Bifurcation analysis of ocean flows using mom4. Ocean Modell. 30, 95–105.
Botta, E.F.F., Wubs, F.W., 1999. Matrix renumbering ILU: an effective algebraic multilevel ILU preconditioner for sparse matrices. SIAM J. Matrix Anal. Appl. 20 (4), 1007–1026.
Cessi, P., Ierley, G.R., 1995. Symmetry-breaking multiple equilibria in quasi-geostrophic, wind-driven flows. J. Phys. Oceanogr. 25, 1196–1205.
De Niet, A.C., Wubs, F.W., Terwisscha van Scheltinga, A.D., Dijkstra, H.A., 2007. A tailored solver for bifurcation studies of ocean-climate models. J. Comp. Phys. 277, 654–679.
Demmel, J.W., Heath, M.T., Heath, M.T., Vorst, H.A.V.D., 1997. Applied numerical linear algebra. In: Society for Industrial and Applied Mathematics. SIAM.
Dijkstra, H.A., 2005. Nonlinear Physical Oceanography: A Dynamical Systems Approach to the Large Scale Ocean Circulation and El Niño, 2nd Revised and Enlarged edition. Springer, New York, pp. 532.
Dijkstra, H.A., Katsman, C.A., 1997. Temporal variability of the wind-driven quasi-geostrophic double gyre ocean circulation: basic bifurcation diagrams. Geophys. Astrophys. Fluid Dyn. 85, 195–232.
Dijkstra, H.A., Weijer, W., 2003. Stability of the global ocean circulation: the connection of equilibria in a hierarchy of models. J. Mar. Res. 61, 725–743.
Dijkstra, H.A., Weijer, W., 2005. Stability of the global ocean circulation: basic bifurcation diagrams. J. Phys. Oceanogr. 35, 933–948.
Jiang, S., Jin, F.-F., Ghil, M., 1995. Multiple equilibria and aperiodic solutions in a wind-driven double-gyre, shallow-water model. J. Phys. Oceanogr. 25, 764–786.
Karypis, G., Kumar, V., 1998. MeTiS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 3.0, Univ. of Minnesota, Dept. of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN, 1998.
Keller, H.B., 1977. Numerical solution of bifurcation and nonlinear eigenvalue problems. In: Rabinowitz, P.H. (Ed.), Applications of Bifurcation Theory. Academic Press, NY, USA.
Knoll, D., Mousseau, V., Chacón, L., Reisner, J., 2005. Jacobian-free Newton–Krylov methods for accurate time integration of stiff wave systems. J. Sci. Comput. 25, 213–229.
Levitus, S., 1994. World Ocean Atlas 1994, Volume 4: Temperature. NOAA/NESDIS E, US Department of Commerce, Washington DC, OC21, pp. 1–117.
Merlis, T.M., Khatiwala, S., 2008. Fast dynamical spin-up of ocean general circulation models using Newton–Krylov methods. Ocean Modell. 21, 97–105.
Morgan, R.B., 2000. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. SIAM J. Matrix Anal. Appl. 21 (4), 1112–1135.
Nadiga, B.T., Taylor, M., Lorenz, J., 2006. Ocean modelling for climate studies: eliminating short time scales in long-term, high-resolution studies of ocean circulation. Math. Comput. Modell. 44 (9–10), 870–886.
Parks, M.L., de Sturler, E., Mackey, G., Johnson, D.D., Maiti, S., 2006. Recycling Krylov subspaces for sequences of linear systems. SIAM J. Sci. Comput. 28 (5), 1651–1674.
Pierini, S., 2006. A Kuroshio extension system model study: decadal chaotic self-sustained oscillations. J. Phys. Oceanogr. 36, 1605–1620.
Primeau, F.W., 1998. Multiple Equilibria and Low-Frequency Variability of Wind-Driven Ocean Models. Ph.D. thesis. M.I.T. and Woods Hole, Boston, MA, USA.
Quon, C., Ghil, M., 1992. Multiple equilibria in thermosolutal convection due to salt-flux boundary conditions. J. Fluid Mech. 245, 449–484.
Quon, C., Ghil, M., 1995. Multiple equilibria and stable oscillations in thermosolutal convection at small aspect ratio. J. Fluid Mech. 291, 33–56.
Rahmstorf, S., 1995. Bifurcations of the Atlantic thermohaline circulation in response to changes in the hydrological cycle. Nature 378, 145–149.
Saad, Y., 1993. A flexible inner–outer preconditioned GMRES algorithm. SIAM J. Sci. Comput 14, 461–469.
Saad, Y., Schultz, M., 1986. A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 7, 856–869.
Schmeits, M.J., Dijkstra, H.A., 2001. Bimodality of the Kuroshio and the Gulf Stream. J. Phys. Oceanogr. 31, 2971–2985.
Simonnet, E., Ghil, M., Dijkstra, H.A., 2005. Homoclinic bifurcations of barotropic QG double-gyre circulation. J. Mar. Res. 63, 931–956.
Simonnet, E., Ghil, M., Ide, K., Temam, R., Wang, S., 2003a. Low-frequency variability in shallow-water models of the wind-driven ocean circulation. Part I: steady-state solutions. J. Phys. Oceanogr. 33, 712–728.
Simonnet, E., Ghil, M., Ide, K., Temam, R., Wang, S., 2003b. Low-frequency variability in shallow-water models of the wind-driven ocean circulation. Part II: time dependent solutions. J. Phys. Oceanogr. 33, 729–752.
Speich, S., Dijkstra, H.A., Ghil, M., 1995. Successive bifurcations of a shallow-water model with applications to the wind driven circulation. Nonlin. Proc. Geophys. 2, 241–268.
Stommel, H., 1961. Thermohaline convection with two stable regimes of flow. Tellus 13, 224–230.
Te Raa, L.A., Dijkstra, H.A., 2002. Instability of the thermohaline ocean circulation on interdecadal time scales. J. Phys. Oceanogr. 32, 138–160.
Thual, O., McWilliams, J.C., 1992. The catastrophe structure of thermohaline convection in a two-dimensional fluid model and a comparison with low-order box models. Geophys. Astrophys. Fluid Dyn. 64, 67–95.
Trenberth, K.E., Olson, J.G., Large, W.G., 1989. A Global Ocean Wind Stress Climatology Based on ECMWF Analyses. Technical Report. National Center for Atmospheric Research, Boulder, CO, USA.
Tuminaro, R.S., Tong, C.H., Shadid, J.N., Devine, K.D., Day, D.M., 2005. Design of a multilevel preconditioning module for unstructured calculations. Num. Lin. Alg. Appl. 1, 369–386.
Van der Vorst, H.A., Vuik, C., 1994. GMRESR: a family of nested GMRES methods. Commun. Num. Methods Eng. 18, 383–389.